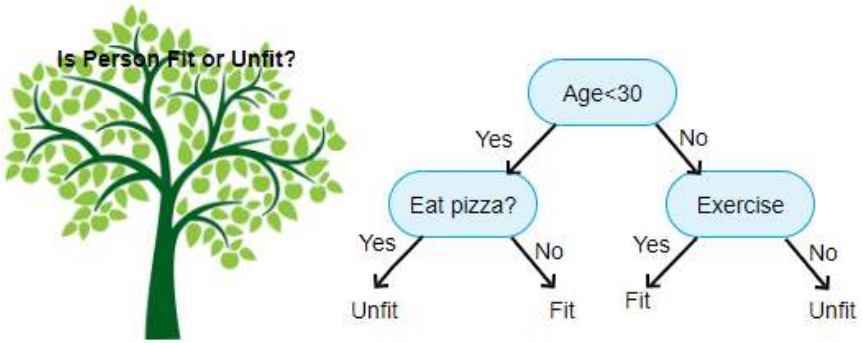
Arbres de décision

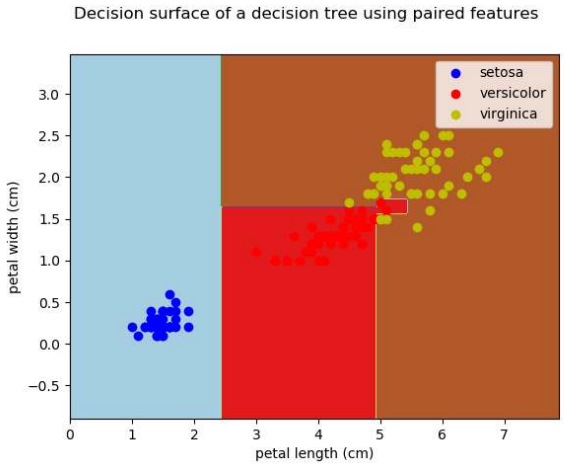


Arbres de décision : Généralités

- Un arbre de décision est un classifieur simple et graphique :
  - Lisibilité
  - Rapidité d'apprentissage et d'exécution
- But :
  - Répartir une population d'individus en groupes homogènes selon un ensemble de variables discriminantes en fonction d'un objectif connu
    $\Longrightarrow$ Apprentissage supervisé
  - Prédire les valeurs prises par la variable à prédire (objectif, variable cible, variable d'intérêt, attribut classe, variable de sortie) à partir d'un ensemble de descripteurs (variables prédictives, variables discriminantes)
    $\Longrightarrow$ régression (variable cible continue)
    $\Longrightarrow$ classification (variable cible discrète)

Arbres de décision

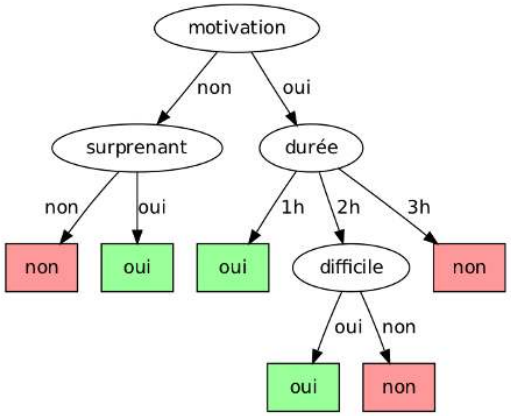
- Une structure de données utilisée comme modèle pour la classification [Quinlan]
- Méthode récursive basée sur l'approche "diviser-pour-régner" pour créer des sous-groupes (plus) purs
  (un sous-groupe est pur lorsque tous les éléments du sous-groupe appartiennent à la même classe)
- Construction du plus petit arbre de décision possible
  - Nœud = Test sur un attribut
  - Une branche pour chaque valeur d'un attribut
  - Les feuilles désignent la classe de l'objet à classer
- Taux d'erreur : proportion des instances qui n'appartiennent pas à la classe majoritaire de la branche
- Problèmes :
  - Choix de l'attribut ?
  - Terminaison ?



Les décisions correspondent à des découpages des données en rectangles

Exemple d'arbre de décision pour la question

« Cette présentation est-elle intéressante ? »



Algorithmes

Les deux algorithmes les plus connus et les plus utilisés sont :

- CART (Classification And Regression Trees [BFOS84])
- C5 (version la plus récente après ID3 et C4.5 [Qui93]).

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International, Monterey, CA, 1984.

[Qui93] J. R. Quinlan. C4.5 : Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

Exemple simple : «jouer au tennis ?»

- Un ensemble de jours (un jour = un exemple)
- Chaque jour caractérisé par un numéro et ses conditions météorologiques (ciel, température, humidité de l'air, force du vent)
- Attribut cible : «jouer au tennis ?» Valeurs possibles : oui et non (classification binaire)

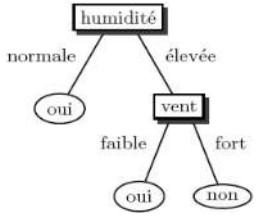
1	Ensoleillé	Chaude	Elevée	Faible	Non
2	Ensoleillé	Chaude	Elevée	Fort	Non
3	Couvert	Chaude	Elevée	Faible	Oui

Exemple - jeu de données

Numéro	Ensoleillement	Temp. (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

Exemple - jeu de données

Numéro	Ensoleillement	Temp. (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui



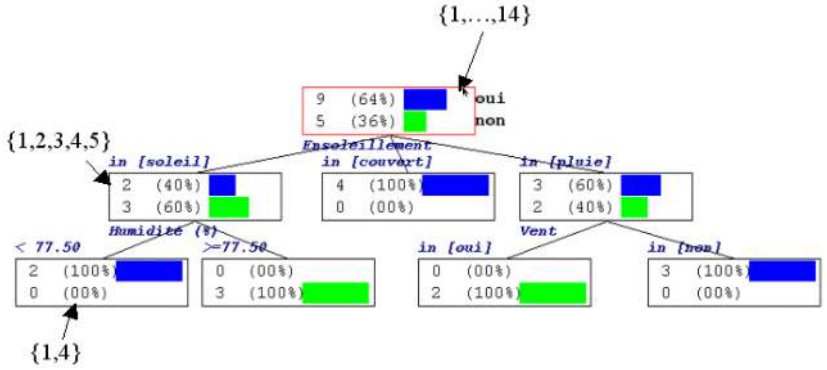
Un exemple d'arbre de décision sur le jeu de données « jouer au tennis ? ».

Exemple simple

- Une fois l'arbre de décision construit, on pourra classer une nouvelle donnée pour savoir si on joue ou non ce jour-là
- Donnée dont la classe est « oui » : positive
- Donnée dont la classe est « non » : négative
- Les classes peuvent correspondre à n'importe quoi : rouge, vert, grand, petit, ...

⇒ Définir adéquatement ce que l'on entend par positif et négatif

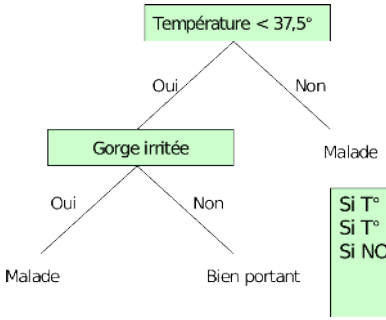
Exemple simple



- Sur les sommets : La distribution de la variable à prédire
- Le premier sommet est segmenté à l'aide de la variable Ensoleillement : 3 sous-groupes ont été produits.
- Le premier groupe comporte 5 observations : 2 correspondent à Jouer = Oui et 3 correspondent à Jouer = Non
- L'arbre peut être traduit en base de règles sans perte d'informations
- Exemple : si ensoleillement = soleil et humidité < 77,5% alors jouer = Oui

Les Arbres de Décision

- Un arbre de décision est une représentation graphique d'une procédure de classification
- Un arbre de décision peut être traduit sous forme de règles de décision
- Exemple :



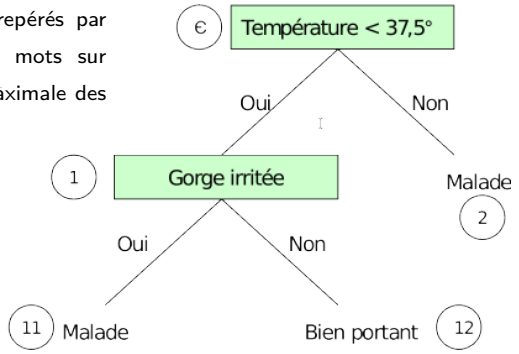
Si T° < 37,5 ET Gorge irritée ALORS Malade
Si T° < 37,5 ET NON(Gorge irritée) ALORS Bien portant
Si NON(T° < 37,5) ALORS Malade

Les Arbres de Décision

- Un arbre de décision est un arbre au sens informatique du terme.
  - Chaque nœud interne teste un attribut
  - Chaque branche correspond à une valeur d'un attribut
  - Chaque nœud feuille est une classe

Les nœuds de l'arbre sont repérés par des positions qui sont des mots sur $\{1, ..., p\}$, où p est l'arité maximale des nœuds.

On note ε le mot vide.



Les Arbres de Décision

Comment générer automatiquement un arbre à partir de données ?

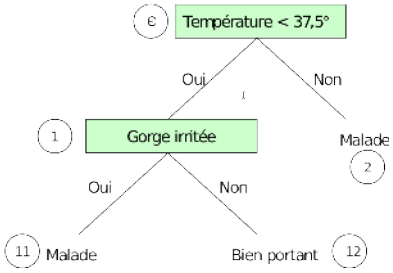
Notations :

- S : échantillon
- $\{1, ..., c\}$: ensemble de classes
- t : arbre de décision
- p : position dans l'arbre
- $N(p)$: cardinal de l'ensemble des exemples associé à p
- $N(k/p)$: cardinal de l'ensemble des exemples associé à p et de classe k
- $P(k/p) = N(k/p)/N(p)$: proportion d'éléments de classe k à la position p

Les Arbres de Décision

	Gorge irritée	Gorge non irritée
T° < 37,5	(6 S, 37 M)	(91 S, 1 M)
T° ≥ 37,5	(2 S, 21 M)	(1 S, 41 M)

M : malade
S : Bien portant



$N(11) = 43$
$N(S/11) = 6$
$N(M/11) = 37$
$P(S/11) = N(S/11)/N(11) = 6/43$
$P(M/11) = N(M/11)/N(11) = 37/43$

Construction d'un arbre de décision

Il existe essentiellement deux familles d'algorithmes :

- les arbres de Quinlan et
- les arbres CART

Lorsque tous les attributs sont discrets avec peu de valeurs différentes

```
1:  ArbreDecision(T)
2:      si "condition d'arret"
3:          retourner feuille(T)
4:      sinon
5:          choisir le "meilleur" attribut i entre 1 et m
6:          pour chaque valeur v de l'attribut i
7:              T[v] = {(x, y) de T tels que x_i = v}
8:              t[v] = ArbreDecision(T[v])
9:          fin pour
10:         retourner noeud(i, {v -> t[v]})
11:     fin si
```

Constructeur pour des attributs à valeurs discrètes

- `noeud(i,{v -> t[v]})` désigne le constructeur d'un nœud qui teste l'attribut i et possède un descendant $t[v]$ pour chaque valeur v possible.

Construction d'un arbre de décision

- Condition d'arrêt influe sur
  - la profondeur
  - la précision du prédicteur produit.

  Par exemple, la condition $|T| = 1 \Rightarrow$ arbres très précis sur l'ensemble d'entraînement (prédiction exacte) MAIS arbres très profonds (longs à calculer)

  $\rightarrow$ risque sur-apprentissage
- Meilleur attribut : il s'agit d'évaluer localement quel attribut apporte « le plus d'information » (ou encore « est le plus corrélé ») au résultat à prédire.
- Lorsque l'attribut x_i est à valeurs réelles, on adapte l'algorithme ci-dessus en choisissant une valeur de partage (split value) v et en effectuant le test $x_i \leq v$. On notera « noeud(i,v,$t_{\leq}$,$t_{>}$) » le constructeur associé. En particulier, si tous les attributs sont réels, l'arbre de décision obtenu est binaire.

Construction d'un arbre de décision

Lorsque tous les attributs sont à valeur dans $\mathbb{R}$

```
1:  ArbreDecision(T)
2:    si "condition d'arret"
3:      retourner feuille(T)
4:    sinon
5:      choisir le "meilleur" attribut i entre 1 et m
6:      choisir une bonne valeur s_i pour l'attribut i
7:        T_<= = {(x, y) de T tels que x_i <= s_i}
7':       T_>  = {(x, y) de T tels que x_i >  s_i}
8:        t_<= = ArbreDecision(T_<=)
8':       t_>  = ArbreDecision(T_>)
9:      fin pour
10:     retourner noeud(i,s_i,v,t_<=,t_>)
11:   fin si
```


Régression

```
1: Regresser(x, t)
2:     si t = feuille(Tf)
3:         retourner la moyenne des y de Tf
4:     sinon si t = noeud(i, v, t_left, t_right)
5:         si x[i] <= v
6:             retourner Regresser(x, t_left)
7:         sinon, x[i] > v
8:             retourner Regresser(x, t_right)
9:     sinon, t = noeud(i, {v -> t[v]})
10:        retourner Regresser(x, t[x[i]])
11:    fin si
```

Régression basée sur un arbre de décision
Certains attributs catégoriels, d'autres continus

Régression : Choix de l'attribut de partage (1)

- CART utilise la variance pour choisir le meilleur attribut de partage (I.5)
- Critères d'homogénéité

$$\text{du nœud } k : \quad D_k = \frac{1}{|k|} \sum_{i \in k} (y_i - \overline{y_k})^2$$

$$\text{des nœuds } k_G \text{ et } k_D : \quad \frac{|k_G|}{|n|} \sum_{i \in k_G} (y_i - \overline{y_{k_G}})^2 + \frac{|k_D|}{|n|} \sum_{i \in k_D} (y_i - \overline{y_{k_D}})^2$$

- Un attribut i produit une partition $T = \cup_{v_i} T_{v_i}$, chaque sous-ensemble ayant sa propre variance $V(T_{v_i})$.
- l'homogénéité attendue après un branchement sur l'attribut i : H_i (pour une instance (x, y) tirée uniformément au hasard dans T) est alors

$$H_i = \sum_{v_i} \frac{|T_{v_i}|}{|T|} (V(T_{v_i}))$$

- l'attribut i^* qui minimise cette valeur est alors considéré (heuristiquement) comme le meilleur choix possible.
- ce choix de minimiser la variance n'est pas seulement heuristique : il est également intrinsèquement lié à la minimisation de l'erreur quadratique de prédiction

Régression : Choix de l'attribut de partage (2)

- Hypothèse du ML : les instances (x_i, y_i) sont tirées indépendamment selon une même loi de probabilité P inconnue.
- On peut alors voir les $(x, y) \in T$ comme les réalisations i.i.d. de variables aléatoires X et Y (éventuellement corrélées).
- De même, les $y \in Y_f$ sont des réalisations i.i.d. d'une certaine variable aléatoire Y_f , qui n'est autre que Y conditionnée par les événements $\{X_i \leq v\}$ ou $\{X_i > v\}$ testés le long de la branche menant à f.
- Dans ce contexte, $\overline{y}$ est un estimateur de $E(Y_f)$ et l'erreur quadratique commise en prédisant $\overline{y}$ pour une nouvelle instance de Y_f s'écrit

$$\begin{aligned} E(Y_f - \overline{y})^2 &= E(Y_f - E(Y_f) + E(Y_f) - \overline{y})^2 \\ &= E(\overline{y} - E(Y_f))^2 + 2(\overline{y} - E(Y_f))E(Y_f - E(Y_f)) + E(Y_f - E(Y_f))^2 \\ &= (\overline{y} - E(Y_f))^2 + E(Y_f - E(Y_f))^2 \end{aligned}$$

($\overline{y}$ et $E(Y_f)$ sont ici constantes)

- $E(Y_f - E(Y_f))^2$: variance σ^2 de Y_f dont $\overline{\sigma^2}$ est un estimateur.
- Minimiser $\overline{\sigma}$ pour la feuille f vise donc à minimiser ce second terme, mais également le premier. En effet, $\overline{y}$ étant une moyenne empirique $\overline{y} = \frac{1}{k}\sum_{i=1}^{k} Y_f(i)$, on a $E_T(\overline{y}) = E(Y_f)$ et $V_T(\overline{y}) = \frac{1}{k}\sigma^2$.

  L'inégalité de Markov (pour Z p.p. positive, $\forall a > 0, \mathbb{P}(Z \geqslant a) \leqslant \frac{\mathbb{E}(Z)}{a}$)

$$P_T[(\overline{y} - E(Y_f))^2 > x] < \frac{\sigma^2}{kx}$$

  $\Rightarrow$ réduire σ a pour effet de concentrer la distribution de $(y - E(Y_f))$ en 0.

Rappel sur la variance

- $Cov(X, Y) = E[(X - E[X])(Y - E[Y])]$ $\quad Var(X) = Cov(X, X)$
- $Var(aX + bY) = a^2 Var(X) + b^2 Var(Y) + 2abCov(X, Y)$
- Ici :

$$\begin{aligned} V_T(\overline{y}) &= Var\left(\frac{1}{k}\sum_{i=1}^{k} Y_f(i)\right) \\ &= \frac{1}{k^2} Var\left(\sum_{i=1}^{k} Y_f(i)\right) \\ &= \frac{1}{k^2}\sum_{i=1}^{k} Var\left(Y_f(i)\right) \\ &= \frac{1}{k}\sigma^2 \end{aligned}$$

Classification

```
1: Classifier(x, t)
2:     si t = feuille(Tf)
3:           retourner la classe majoritaire de Tf
4:     sinon si t = noeud(i, v, t_left, t_right)
5:            si x[i] <= v
6:                  retourner Classifier(x, t_left)
7:            sinon, x[i] > v
8:                  retourner Classifier(x, t_right)
9:     sinon, t = noeud(i, {v -> t[v]})
10:           retourner Classifier(x, t[x[i]])
11:    fin si
```

Classification basée sur un arbre de décision
Certains attributs catégoriels, d'autres continus

Classification : Mesurer l'homogénéité des feuilles

Mesurer le l'homogénéité $\equiv$ mesurer le désordre

- Soient $p_1, ..., p_C$ les fréquences relatives des classes 1,...,C dans T_f , et c^* la classe la plus fréquente.
- **Taux d'erreur :** $e(T_f) := 1 - p_{c^*}$
  Il s'agit du taux d'erreurs de classification sur l'ensemble d'entraînement.
- **Critère de Gini :** $e(T_f) = \sum_c p_c(1 - p_c)$
  Il s'agit du taux d'erreur sur l'ensemble d'entraînement d'un algorithme randomisé qui retournerait la classe c avec probabilité p_c (au lieu de toujours retourner la classe c^*). C'est la mesure d'erreur utilisée dans CART.
- **Entropie :** $e(T_f) = -\sum_c p_c log(p_c)$
  Il s'agit d'un estimateur de l'entropie de la classe d'une instance de T_f tirée uniformément au hasard. C'est la mesure d'erreur utilisée dans les arbres ID3 et C4.5.

Classification : Choix de l'attribut de partage

- La mesure d'erreur est utilisée pour choisir le meilleur attribut de partage : si un attribut i partitionne T en $T = \cup_{v_i} T_{v_i}$, chaque ensemble T_{v_i} a sa propre erreur $e(T_{v_i})$ et l'erreur attendue après un branchement sur cet attribut (pour un élément tiré uniformément au hasard de T) est

$$e_i = \sum_{v_i} \frac{|T_{v_i}|}{|T|} e(T_{v_i}).$$

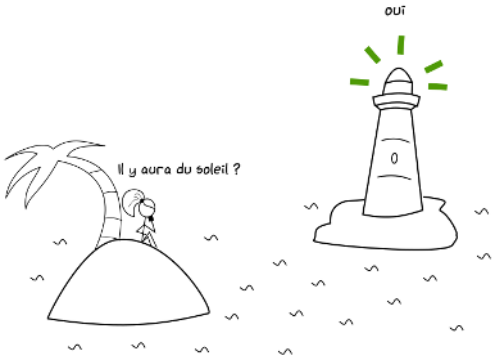
L'attribut qui minimise cette erreur est (heuristiquement) considéré comme le meilleur choix possible.

Classification : Entropie (ID3, C4.5)

Entropie : Une mesure de l'incertitude

- Entropie : mesure de l'incertitude d'un événement en fonction de la connaissance que nous avons.
- Le soleil se lève tous les jours : on est donc certain qu'il se lèvera demain.
- Croiser un chat noir : Cela m'est déjà arrivé plusieurs fois, mais rien ne garantit que cela arrive aujourd'hui.

⇒ Pour lever cette incertitude, je dois récupérer une certaine quantité d'information...

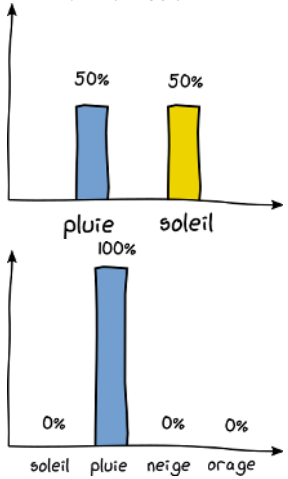


- un téléphone pour contacter le gardien d'un phare.
- pendant un 1 mois, prévision météo du jour.
- le micro du gardien casse (il entend, mais ne peut pas répondre)
- sa réponse : oui/non en utilisant le signal lumineux de son phare : verte (Oui), rouge (non).
- Combien de questions au minimum allez-vous poser au gardien du phare pour lever l'incertitude sur la météo du jour ?

Entropie : Une mesure de l'incertitude

- Entropie : mesure de l'incertitude d'un événement en fonction de la connaissance que nous avons.
- Le soleil se lève tous les jours : on est donc certain qu'il se lèvera demain.
- Croiser un chat noir : Cela m'est déjà arrivé plusieurs fois, mais rien ne garantit que cela arrive aujourd'hui.

⇒ Pour lever cette incertitude, je dois récupérer une certaine quantité d'information...



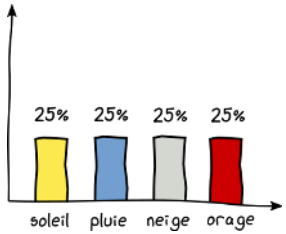
Est-ce qu'il va pleuvoir aujourd'hui ?
le phare vous a envoyé 1 bit de donnée
L'incertitude de 1 chance sur 2 a été divisée par 2.

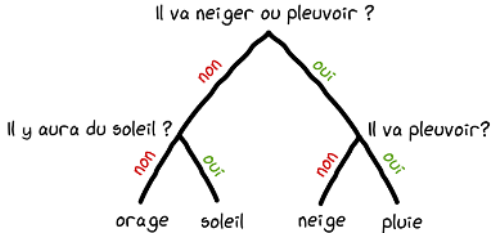
Aucune question
le phare vous a envoyé 0 bit de donnée
l'incertitude est nulle.

Entropie : Une mesure de l'incertitude

- Entropie : mesure de l'incertitude d'un événement en fonction de la connaissance que nous avons.
- Le soleil se lève tous les jours : on est donc certain qu'il se lèvera demain.
- Croiser un chat noir : Cela m'est déjà arrivé plusieurs fois, mais rien ne garantit que cela arrive aujourd'hui.

⇒ Pour lever cette incertitude, je dois récupérer une certaine quantité d'information...





le phare vous a envoyé 2 bit de donnée
l'incertitude de 1 chance sur 4 a été divisée par 4.

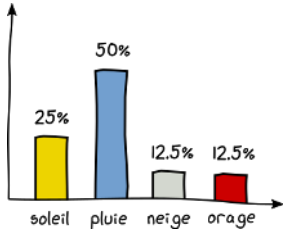
```
vert-vert (11) = pluie
vert-rouge (10) = neige
rouge-vert (01) = soleil
rouge-rouge (00) = orage
```


CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Généralités
Algorithmes
Régression
Classification
Extensions
Conclusion
Clustering
kNN
Bayes
SVM
NN
Evaluation
Data Mining
Project

Entropie : Une mesure de l'incertitude

- Entropie : mesure de l'incertitude d'un événement en fonction de la connaissance que nous avons.
- Le soleil se lève tous les jours : on est donc certain qu'il se lèvera demain.
- Croiser un chat noir : Cela m'est déjà arrivé plusieurs fois, mais rien ne garantit que cela arrive aujourd'hui.

⇒ Pour lever cette incertitude, je dois récupérer une certaine quantité d'information...



le phare vous a envoyé 1,2 ou 3 bit de données
l'incertitude de 1 chance sur 4 a été divisée par 4.

```
vert (1) = pluie
rouge-vert (01) = soleil
rouge-rouge-vert (001) = neige
rouge-rouge-rouge (000) = orage
```


CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Généralités
Algorithmes
Régression
Classification
Extensions
Conclusion
Clustering
kNN
Bayes
SVM
NN
Evaluation
Data Mining
Project

Classification : Entropie (ID3, C4.5)

- Interprétation :
  - quantité d'information délivrée par une source d'information
  - Pour un récepteur, plus la source émet d'informations différentes, plus l'entropie (ou incertitude sur ce que la source émet) est grande.
  - si la source envoie toujours le même symbole *a*, alors son entropie est nulle, c'est-à-dire minimale.
    le récepteur peut prédire que le prochain symbole sera un *a*.
  - si la source envoie *a* la moitié du temps et *b* l'autre moitié, le récepteur est incertain de la prochaine lettre à recevoir.
    L'entropie de la source est alors non-nulle (positive) et représente l'incertitude qui règne sur l'information émanant de la source.
  - L'entropie indique alors la quantité d'information nécessaire pour que le récepteur puisse déterminer sans ambiguïté ce que la source a transmis.
    plus la source est redondante, moins elle contient d'information.
    En l'absence de contraintes particulières, l'entropie est maximale pour une source dont tous les symboles sont équiprobables.
  - ⇒ Dans notre cas : nombre minimum de bits nécessaires pour coder la classe d'un élément quelconque

CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Généralités
Algorithmes
Régression
Classification
Extensions
Conclusion
Clustering
kNN
Bayes
SVM
NN
Evaluation
Data Mining
Project

Classification : Entropie (ID3, C4.5)

- Combien d'information me porte un message ?
  - « Le ciel est bleu », « la terre est ronde » - pas d'information en termes de quantité : information nulle
  - La quantité d'information est liée à l'effet surprise : plus un événement est improbable, plus le fait de recevoir une information le concernant nous apporte de l'information
  - Lancer d'une pièce de monnaie et un dé à six cotés. Quelqu'un nous dit le résultat. Quelle information est plus précieuse ?
- Difficulté pour « deviner » le résultat

Classification : Entropie $H(X) = -\sum P(x_i)log(x_i)$

- Considérons un ensemble X d'exemples dont une proportion p^+ sont positifs et une proportion p^- sont négatifs. ($p^+ + p^- = 1$). L'entropie de X est :

$$H(X) = -(p^+)log_2(p^+) - (p^-)log_2(p^-)$$

- Remarques (deux proportions)
  - $0 \leq H(X) \leq 1$
  - Si $p^+ = 0$ ou $p^- = 0$, alors $H(X) = 0$
  - Si $p^+ = p^- = 0,5$, alors $H(X) = 1$ (entropie maximale) : autant de positifs que de négatifs
- Cas général (N proportions)
  Pour un attribut classe prenant N valeurs distinctes ; $p_i (i \in \{1, N\})$, la proportion d'exemples dont la valeur de cet attribut est i dans l'ensemble d'exemples X, L'entropie de l'ensemble d'exemples X est :

$$H(X) = -(p_1 log_2 p_1 + p_2 log_2 p_2 + ... + p_N log_2 p_N)$$

Implémentation : ID3 (Iterative Dichotomiser 3)

- On suppose que la variable cible a m valeurs distinctes (les étiquettes de classe). Pour un nœud S (interne ou feuille) on calcule son entropie par rapport à la cible comme suit :
  - Partitionner S sur les valeurs de la cible en m groupes : $C_1, ..., C_m$
  - Calculer $p_i, i = 1...m$, la probabilité qu'un élément de S se retrouve dans C_i ($p_i \equiv |C_i|/|S|$ où $|C_i|$ est la taille du groupe C_i)
  - $H(S) = -\sum_i p_i \log(p_i)$ est l'entropie de S.
- $H(S)$ mesure l'écart de la distribution de la variable cible par rapport à la distribution uniforme :
  - $H(S) = 0$ si S est homogène : toutes les probabilités pi sont égales à 0, sauf une qui est égale à 1
  - $H(S) = max$ si toutes les probabilités p_i sont égales (tous les groupes C_i ont la même taille : $p_1 = ... = p_n = 1/m$).
- Pour calculer le gain d'information dans un nœud interne S sur l'attribut a :
  - Partitionner S sur les valeurs de l'attribut a en k sous-groupes : $S_1, ..., S_k$ (k est le nombre de valeurs distinctes de l'attribut a),
  - p_i : la probabilité qu'un élément de S appartient à S_i ($p_i \equiv |S_i|/|S|$)
  - Calculer $GI(S, a) = H(S) - \sum_{i=1}^{k} p_i H(S_i)$ le gain d'information sur l'attribut a.

Implémentation : ID3 (Iterative Dichotomiser 3)

- l'algorithme ID3 commence par la racine. Ensuite pour le nœud S en train d'être testé :
  - Calculer le gain d'information pour chaque attribut pas encore utilisé,
  - Choisir l'attribut a de gain d'information maximal,
  - Créer un test (décision) sur cet attribut dans le nœud S et générer les sous-nœuds $S_1, ...S_k$ correspondant à la partition sur l'attribut choisi a
  - Récurrence sur les nœuds qui viennent d'être crées.
- Sortie de la récursivité :
  - Tous les éléments de S sont dans la même classe ($H(S) = 0$) : S devient une feuille,
  - Plus d'attributs non utilisés : nœud feuille sur le classe majoritaire,
  - $S = \emptyset$: nœud feuille sur le classe majoritaire du parent (ce cas est nécessaire pour le classement de nouveaux échantillons).

exemple

Exemple *[Quinlan,86]*

Attributs Valeurs possibles	Pif soleil,couvert,pluie	Temp chaud,bon,frais	Humid normale,haute	Vent vrai,faux

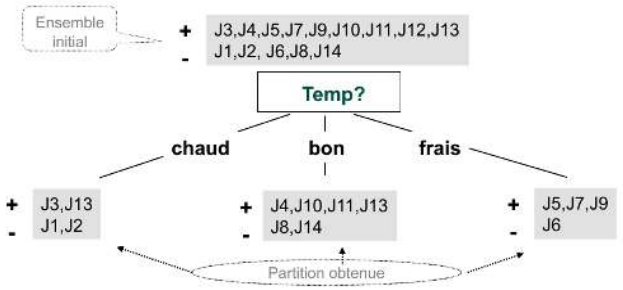
N°	Pif	Temp	Humid	Vent	Golf
1	soleil	chaud	haute	faux	NePasJouer
2	soleil	chaud	haute	vrai	NePasJouer
3	couvert	chaud	haute	faux	Jouer
4	pluie	bon	haute	faux	Jouer
5	pluie	frais	normale	faux	Jouer
6	pluie	frais	normale	vrai	NePasJouer
7	couvert	frais	normale	vrai	Jouer
8	soleil	bon	haute	faux	NePasJouer
9	soleil	frais	normale	faux	Jouer
10	pluie	bon	normale	faux	Jouer
11	soleil	bon	normale	vrai	Jouer
12	couvert	bon	haute	vrai	Jouer
13	couvert	chaud	normale	faux	Jouer
14	pluie	bon	haute	vrai	NePasJouer

la classe

Développement de l'arbres de décision : exemple

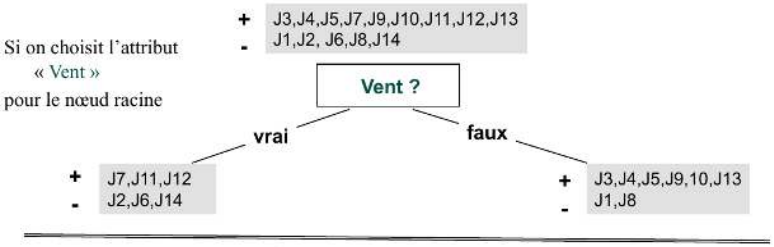
Si on choisit l'attribut « Temp »
pour le nœud racine

N°	Pif	Temp	Humid	Vent	Golf
1	soleil	chaud	haute	faux	NePasJoue
2	soleil	chaud	haute	vrai	NePasJoue
3	couvert	chaud	haute	faux	Jouer
4	pluie	bon	haute	faux	Jouer
5	pluie	frais	normale	faux	Jouer
6	pluie	frais	normale	vrai	NePasJoue
7	couvert	frais	normale	vrai	Jouer
8	soleil	bon	haute	faux	NePasJoue
9	soleil	frais	normale	faux	Jouer
10	pluie	bon	normale	faux	Jouer
11	soleil	bon	normale	vrai	Jouer
12	couvert	bon	haute	vrai	Jouer
13	couvert	chaud	normale	faux	Jouer
14	pluie	bon	haute	vrai	NePasJoue

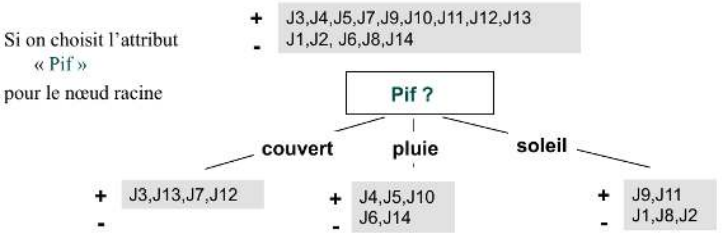


Induction d'arbres de décision : sélection de l'attribut

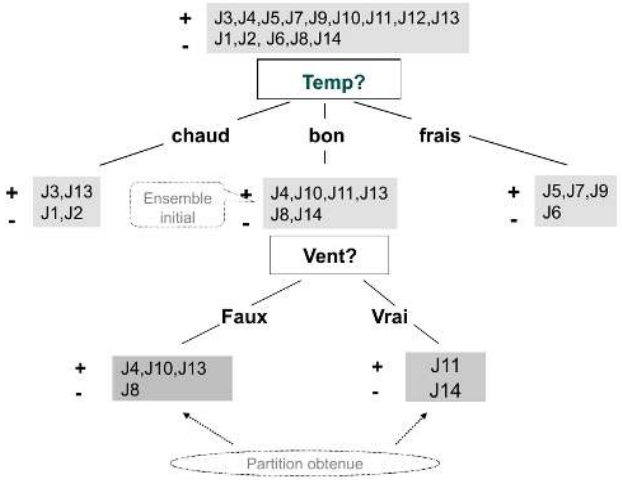
Si on choisit l'attribut
« Vent »
pour le nœud racine



Si on choisit l'attribut
« Pif »
pour le nœud racine



Développement de l'arbre à partir d'un noeud pendant (feuille)
Exemple



3- Exemple

- Entropie de l'ensemble initial d'exemples

  $I(p,n) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14)$

- Entropie des sous-arbres associés au test sur Pif ?
  - $p_1 = 4 \quad n_1 = 0 : \quad I(p_1,n_1) = 0$
  - $p_2 = 2 \quad n_2 = 3 : \quad I(p_2,n_2) = 0.971$
  - $p_3 = 3 \quad n_3 = 2 : \quad I(p_3,n_3) = 0.971$
- Entropie des sous-arbres associés au test sur Temp ?
  - $p_1 = 2 \quad n_1 = 2 : \quad I(p_1,n_1) = 1$
  - $p_2 = 4 \quad n_2 = 2 : \quad I(p_2,n_2) = 0.918$
  - $p_3 = 3 \quad n_3 = 1 : \quad I(p_3,n_3) = 0.811$

exemple

- Pour les exemples initiaux

  $I(S) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14)$

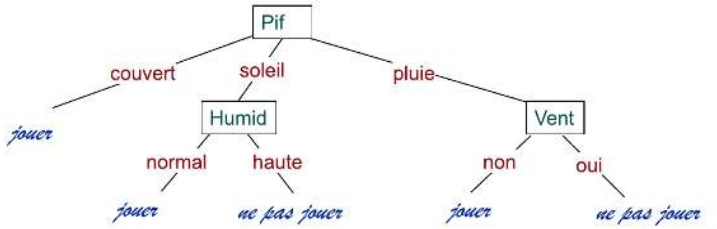
- Entropie de l'arbre associé au test sur Pif ?
  - $E(\text{Pif}) = 4/14\ I(p_1,n_1) + 5/14\ I(p_2,n_2) + 5/14\ I(p_3,n_3)$

    $\text{Gain(Pif)} = 0.940 - 0.694 = 0.246$ bits
  - $\text{Gain(Temp)} = 0.029$ bits
  - $\text{Gain(Humid)} = 0.151$ bits
  - $\text{Gain(Vent)} = 0.048$ bits

Choix de l'attribut Pif pour le premier test

exemple

- Arbre final obtenu :



CART : Breiman et al. [BF84]

Arbres de classification et régression

- Les principales différences sont les suivantes :
  - CART pose seulement de questions-test binaires (arbres binaires).
  - Fonctionne aussi pour des attributs aux valeurs continues.
  - CART cherche tous les attributs et tous les seuils pour trouver celui qui donne la meilleure homogénéité du découpage.
- Quand un nœud interne S est coupé sur l'attribut j, seuil a_j, il donne naissance à deux descendants :
  - Sous-nœud gauche S_g ($p_g \equiv |S_g|/|S|$) qui contient tous les éléments qui ont les valeurs de l'attribut $v_j \leq a_j$
  - Sous-nœud droit S_d ($p_d \equiv |S_d|/|S|$) qui contient tous les éléments qui ont les valeurs de l'attribut $v_j > a_j$.
- Soit $I(S)$ une mesure de l'impureté de S par rapport à la classe cible. CART étudie le changement de l'impureté par rapport au seuil et pour tous les attributs :
  - $E[I(S_{gd})] = p_g I(S_g) + p_d I(S_d)$ ou $E[.]$ est l'opérateur ésperance statistique,
  - $\Delta I(S) = I(S) - E[I(S_{gd})] = I(S) - p_g I(S_g) - p_d I(S_d)$

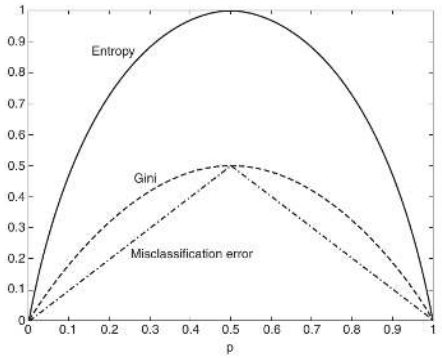
CART : Le problème d'optimisation

- Le problème d'optimisation est le suivant :

$$\underset{j;a_j}{argmax}\ \Delta I(S)$$

  CART choisit donc (j, a_j) qui maximise la décroissance de l'impureté du nœud par rapport à la cible.
- En classification la mesure de l'impureté utilisée est l'index (ou impureté) de Gini qui est la vraisemblance qu'un élément du nœud soit incorrectement étiquetté par un tirage aléatoire qui respecte la loi statistique de la cible estimée dans le nœud.
- L'impureté (ou l'index de Gini $I_G(S)$) pour un nœud S est calculée comme suit :
  - Partitionner S sur les valeurs de la cible en n groupes : $C_1, ..., C_n$,
  - Calculer p_i : probabilité estimée qu'un élément de S se retrouve dans C_i ($p_i \equiv |C_i|/|S|$),
  - $I_G(S) = \sum_{i=1}^{m} p_i(1-p_i) = \sum_{i=1}^{m}(p_i - p_i^2) = 1 - \sum_{i=1}^{m} p_i^2$,
  - $I_G(S) = \sum_{i \neq j} p_i p_j$ index de Gini,
  - $I_G(S) = 0$ si S est homogène (tous les éléments sont dans la même classe, donc impureté du groupe nulle).
- Toujours en classification on peut utiliser d'autres types de mesures d'impureté :
  - $H(s) = -\sum_i p_i \log(p_i)$ (entropie),
  - $E(s) = 1 - max_i p_i$ (erreur de classification).

CART : Le problème d'optimisation



Pour un problème de régression on optimise le résidu quadratique moyen : minimise la variance moyenne des groupes.

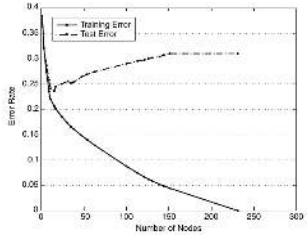
$$\underset{j;a_j}{argmin}\, p_g Var(S_g) + p_d Var(S_d)$$

Extension des arbres de décision

Plusieurs limitations :

- le problème d'optimisation global est NP-complet [6] pour de nombreux critères d'optimalité, conduisant à l'emploi d'heuristiques ;
- la procédure d'apprentissage est statique : ils ne sont pas prévus pour apprendre de manière incrémentale de nouvelles instances qui viendraient s'ajouter à l'ensemble d'entraînement ;
- ils sont sensibles au bruit et ont une forte tendance à sur-apprendre les données, *i.e.* à apprendre à la fois les relations entre les données et le bruit présent dans l'ensemble d'apprentissage.

Solutions :



- l'élaguage
- « bagger » les arbres plutôt que de les utiliser comme prédicteurs individuels, un contexte dans lequel le sur-apprentissage et la sous-optimalité dû aux heuristiques posent moins de problèmes.

Bagging

Le « bagging », acronyme pour « **b**ootstrap **agg**regat**ing** », est un méta-algorithme qui combine deux techniques

- **Bootstrapping :** un bootstrap d'un ensemble T est l'ensemble obtenu en tirant $|T|$ fois des éléments de T uniformément au hasard et avec remise. Le bootstrapping d'un ensemble d'entraînement T produit un nouvel ensemble T_0 qui présente en moyenne $1 - e^{-1} \equiv 63\%$ instances uniques différentes de T quand $|T| >> 1$.
- **Aggrégation :** on produit plusieurs bootstraps $T_1, ..., T_m$, chaque bootstrap T_i étant utilisé pour entraîner un prédicteur t_i (penser ici à un arbre de régression, mais la technique s'applique à n'importe quelle famille de prédicteurs). Étant donnée une instance (x, y), on fait régresser chaque arbre, ce qui nous donne un ensemble de valeurs $y_1, ..., y_m$ prédites. Celles-ci sont alors aggrégées en calculant leur moyenne $\overline{y} = \frac{1}{m} \sum_i y_i$.

Bagging

Le bagging corrige plusieurs défauts des arbres de décision :

- leur instabilité (de petites modifications dans l'ensemble d'apprentissage peuvent entraîner des arbres très différents) et
- leur tendance à surapprendre.

La contrepartie à payer est une perte de lisibilité :

- les prédictions d'une forêt d'arbres « baggés » ne sont plus le fruit d'un raisonnement, mais
- un consensus de raisonnements potentiellement très différents.

L'analyse théorique du bagging demeure incomplète à ce jour : plusieurs arguments aident à comprendre son impact et suggèrent des conditions dans lesquelles il améliore les prédictions, mais l'élaboration d'un modèle dans lequel cet impact est compris et mesurable reste un sujet de recherche.

Random Forests

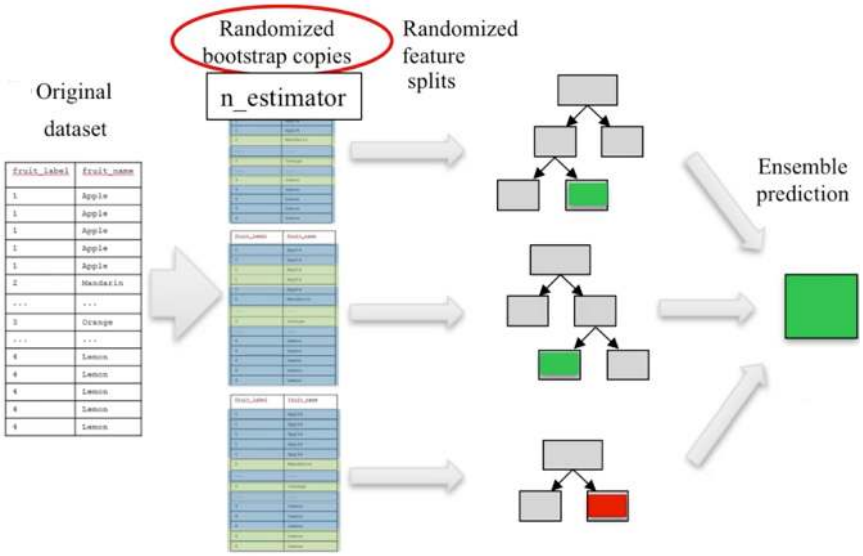
Proposées en 2002 par Leo Breiman et Adele Cutler, les forêts aléatoires (Random Forests) modifient l'algorithme pour que les arbres construits soient adaptés au bagging. Les principales modifications sont les suivantes :

- **Échantillonage des attributs :** à la ligne 5 de l'algorithme, au lieu de choisir l'attribut i dans l'ensemble $A = \{1, ..., m\}$, on tire uniformément au hasard un sous-ensemble $A' \subset A$ de taille $m' \leq m$ (pour la régression, les auteurs suggèrent $m' \equiv m/3$) dont on choisira le meilleur attribut.

  L'objectif de cette opération est de décorréler les arbres produits, la corrélation entre les prédicteurs réduisant les gains potentiels en précision dus au bagging.

- **Critère d'arrêt :** les arbres construits sont aussi profonds que possible, *i.e.* le critère d'arrêt est une condition forte ($|T_f| < 10, |Y_f| = 1, ...$), dans les limites d'un temps de calcul raisonnable. Ainsi, un prédicteur apprendra toutes les relations entre les données à sa portée, le sur-apprentissage et la sensibilité au bruit qui en résultent étant compensés par le bagging.

Random Forests



Arbre de décision : Conclusion

- Méthodes non paramétrique pour classification + régression
- Données catégorielles + données numériques
- avantages :
  1. simples à comprendre et à visualiser.
  2. peu de préparation des données (normalisation, etc.).
  3. Le coût d'utilisation des arbres est logarithmique.
  4. capables de traiter des problèmes multi-classes.
  5. White box : le résultat facile à conceptualiser / visualiser.
- Désavantages :
  1. Sur-apprentissage : arbres parfois trop complexes ⇒ mauvaise généralisation. Choisir des bonnes valeurs pour max_depth et min_samples_leaf
  2. Arbres parfois déséquilibrés (⇒ temps de parcours plus logarithmique). Ajuster la base de données en amont pour éviter qu'une classe domine largement les autres (en termes de nombre d'exemples d'apprentissage).

Under and Overfitting

- Factors
  - Main : model complexity/flexibility (also degrees of freedom)
  - Second : tune model parameters to optimize performances on learning set
- Illustration
  - Linear : not enough flexible ⇒ underfitting
    Consequences :
    - Not that good on learning set
    - Similar performances on new data
  - Quadratic : true model
  - Higher-order polynomial : too flexible ⇒ overfitting
    Consequences :
    - Good on leaning set
    - Not that good on new data
- Conclusions
  - Find a complexity trade-off
  - Do not rely on performances on (full) learning set